R-Skript PUNO-Forschungsprojekt

Teil 2 – Bi- und multivariate Statistik

Dominik Vogel

Stand: 03.04.2019

Inhaltsverzeichnis

1 Pakete

Wir benötigen zunächst das `tidyverse` Paket, das wir schon aus Teil 1 kennen.

```
library(tidyverse)
```

2 Bivariate Statistik

Wir wollen uns natürlich nicht nur Verteilungen und Mittelwerte anschauen, sondern Zusammenhänge untersuchen. Starten wir also mit Zusammenhängen zwischen zwei Variablen. Wir widmen uns dabei vier Verfahren:

- *t*-Test
- ANOVA
- Korrelation
- bivariate Regression

2.1 *t*-Test

Der *t*-Test kommt in verschiedenen Formen vor:

- Einstichproben-*t*-Test (One-sample *t*-test)
- Zweistichproben-*t*-Test (Two-sample *t*-test / independent *t*-test)
- Abhängiger *t*-Test (Paardifferenzentest / paired *t*-test / dependent *t*-test)

> „Die t-Statistik wurde im Jahre 1908 von William Sealy Gosset eingeführt. Er arbeitete als Chemiker für die Guinness-Brauerei in Dublin (Irland) und entwickelte den t-Test als eine billige Art und Weise, die Qualität des Stout zu überwachen. Guinness verbot seinen Mitarbeitern, Ergebnisse zu publizieren, daher veröffentlichte Gosset seine Arbeit unter dem Pseudonym Student.“ (https://de.wikipedia.org/wiki/T-Test)

In diesem Kapitel verwenden wir zwei Datensätze von Field et al. (2012) zu Arachnophobie (Angst vor Spinnen). In beiden Datensätzen wurden Probanden mit Arachnophobie einer Spinne ausgesetzt – einmal tatsächlich und einmal auf einem Bild. Anschließend wurde ihr Angstlevel gemessen. Die Datensätze unterscheiden sich lediglich in einem Aspekt: `spider_long` ist das Resultat eines between-groups Designs (jeder Proband erhielt nur eines von zwei Treatments) und `spider_wide` eines within-person Designs (jeder Proband erhielt beide Treatments).

```
spider_long <- read_csv("data/spider_long.csv")
```

```
## Parsed with column specification:
## cols(
##   group = col_character(),
##   anxiety = col_double()
## )
```

```
spider_long
```

```
## # A tibble: 24 x 2
##    group   anxiety
##    <chr>     <dbl>
##  1 Picture      30
##  2 Picture      35
##  3 Picture      45
##  4 Picture      40
##  5 Picture      50
##  6 Picture      35
##  7 Picture      55
##  8 Picture      25
##  9 Picture      30
## 10 Picture      45
## # ... with 14 more rows
```

```r
spider_wide <- read_csv("data/spider_wide.csv")
```

```
## Parsed with column specification:
## cols(
##   picture = col_double(),
##   real = col_double()
## )
```

```r
spider_wide
```

```
## # A tibble: 12 x 2
##    picture  real
##      <dbl> <dbl>
##  1      30    40
##  2      35    35
##  3      45    50
##  4      40    55
##  5      50    65
##  6      35    55
##  7      55    50
##  8      25    35
##  9      30    30
## 10      45    50
## 11      40    60
## 12      50    39
```

2.1.1 Einstichproben-*t*-Test

Der Einstichproben-*t*-Test prüft, ob der Mittelwert einer Stichprobe, einem bestimmten Wert entspricht. Wir können beispielsweise testen, ob der mittlere Anxiety Score signifikant von 40 abweicht.

$$Nullhypothese\ H_0 : \mu = 40$$

$$Alternativhypothese\ H_a : \mu \neq 40$$

```r
t.test(spider_long$anxiety, mu = 40)
```

```
##
##  One Sample t-test
##
## data:  spider_long$anxiety
## t = 1.6183, df = 23, p-value = 0.1192
## alternative hypothesis: true mean is not equal to 40
## 95 percent confidence interval:
##  39.02599 47.97401
## sample estimates:
## mean of x
##      43.5
```

Der tatsächliche Mittelwert beträgt 43,5. Der p-Wert für die Alternativhypothese ($Mittelwert \neq 40$) liegt bei 0,12. Die Nullhypothese kann also nicht verworfen werden. Der Mittelwert unserer Stichprobe ist nicht signifikant von 40 verschieden.

Wir können mit dem Einstichproben-t-Test auch prüfen, ob der Mittelwert über oder unter einem bestimmten Wert liegt. Hierfür setzen wir die Option `alternative` auf `"greater"` oder `"less"`.

```
t.test(spider_long$anxiety, mu = 35, alternative = "greater")
```

```
##
##  One Sample t-test
##
## data:  spider_long$anxiety
## t = 3.9302, df = 23, p-value = 0.0003345
## alternative hypothesis: true mean is greater than 35
## 95 percent confidence interval:
##  39.79331      Inf
## sample estimates:
## mean of x
##      43.5
```

Hier haben wir nun ein signifikantes Ergebnis. Die Nullhypothese (Mittelwert ist kleiner gleich 35) wird verworfen. Wir finden also Unterstützung für die Alternativhypothese (Mittelwert ist größer 35).

Annahmen für den Einstichproben-t-Test: Normalverteilung (s. unten), unabhängige Beobachtungen

2.1.2 Zweistichproben-t-Test

Interessanter als der Einstichproben-t-Test, ist der Zweistichproben-t-Test. Er vergleicht den Mittelwert zweier Gruppen miteinander.

$$Nullhypothese\ H_0 : \mu_{picture} = \mu_{realspider}$$

$$Alternativhypothese\ H_a : \mu_{picture} \neq \mu_{realspider}$$

Wir nutzen wieder `t.test`. Dieses Mal geben wir aber keinen Mittelwert, sondern die Gruppenvariable an. Abhängige Variable und Gruppe werden mit einem `~` getrennt: `anxiety ~ group`. mit `paired = FALSE` geben wir schließlich noch an, dass es sich um unabhängige Beobachtungen handelt.

```
t.test(anxiety ~ group, data = spider_long, paired = FALSE)
```

```
##
##  Welch Two Sample t-test
##
## data:  anxiety by group
## t = -1.6813, df = 21.385, p-value = 0.1072
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -15.648641   1.648641
## sample estimates:
##     mean in group Picture mean in group Real Spider
##                        40                        47
```

Die Nullhypothese kann also nicht verworfen werden (p = 0,107). Man berichtet das Ergebnis eines solchen Tests übrigens (nach den Standards der APA) folgendermaßen: $t(21.39) = -1.68$, $p = .107$. In die Klammer hinter das t kommen die Freiheitsgrade (df), gefolgt vom t-Wert und dem p-Wert.

2.1.2.1 Effektstärke

Den Output des t-Tests können wir auch nutzen, um die Effektstärke zu berechnen. Für die Unterschiede zwischen zwei Gruppen verwendet man gewöhnlich Cohen's d. Dieses gibt an, wie groß der Unterschied zwischen den zwei Mittelwerten im Vergleich zur Standardabweichung ist. Die Formel sieht so aus:

$$d = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{(s_1^2 + s_2^2)/2}}$$

$\bar{x}_1$ und $\bar{x}_2$ sind die Mittelwerte der beiden Gruppen. s_1 und s_2 sind die Standardabweichungen der beiden Gruppen.

Damit wir das nicht von Hand berechnen müssen, verwenden wir `cohen.d()` aus dem Paket `effsize`.

```
install.packages("effsize", dep = TRUE)
```

```
library(effsize)
```

```
cohen.d(anxiety ~ group, data = spider_long)
```

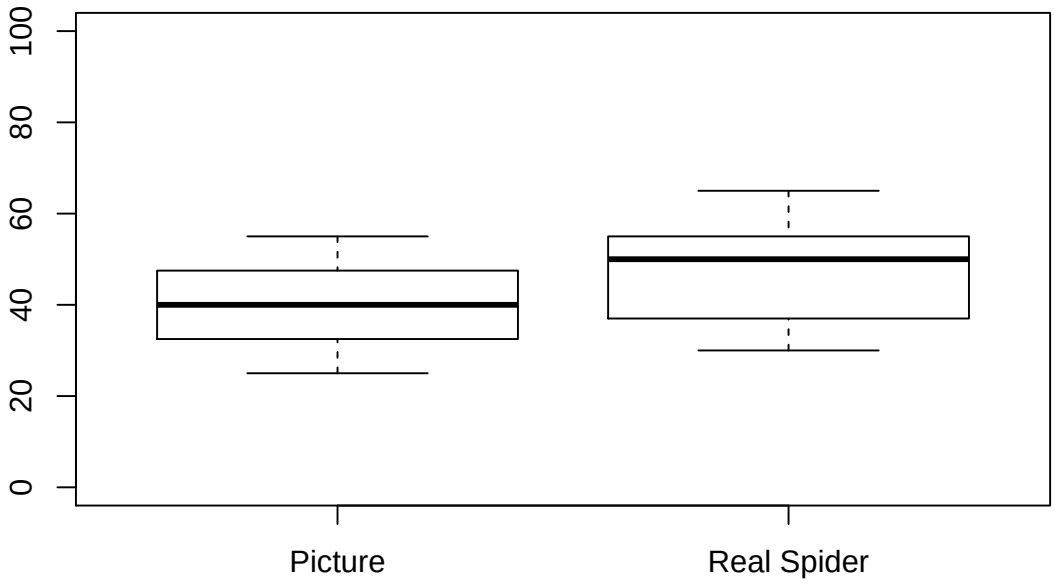
```
##
## Cohen's d
##
## d estimate: -0.6864065 (medium)
## 95 percent confidence interval:
##      lower      upper
## -1.5576365  0.1848236
```

Es gibt vielfältige Empfehlungen, wie diese Werte zu interpretieren sind. Oftmals wird ein Effekt von 0,2 bis 0,5 als *klein*, von 0,5 bis 0,8 als *mittel* und bei größer 0,8 als *groß* betrachtet. Letztlich kommt es aber auf Ihre Interpretation an.

Die Mittelwerte der beiden Gruppen sind also nicht signifikant verschieden voneinander. Wir sollten also nicht davon ausgehen, dass wir einen Effekt gefunden haben. Wenn unserem kleinen Experiment eine gute Theorie zugrunde liegt, lohnt es sich aber vermutlich ein erneutes Experiment mit einer größeren Fallzahl (als mehr Power) durchzuführen. Das sollten wir aber aufgrund der kleinen Stichprobe in diesem Experiment weder aus dem p-Wert noch aus der Effektstärke ableiten.

Den deutlichen Unterschied können wir auch grafisch darstellen:

```
boxplot(spider_long$anxiety ~ spider_long$group,
        ylim = c(0, 100))
```

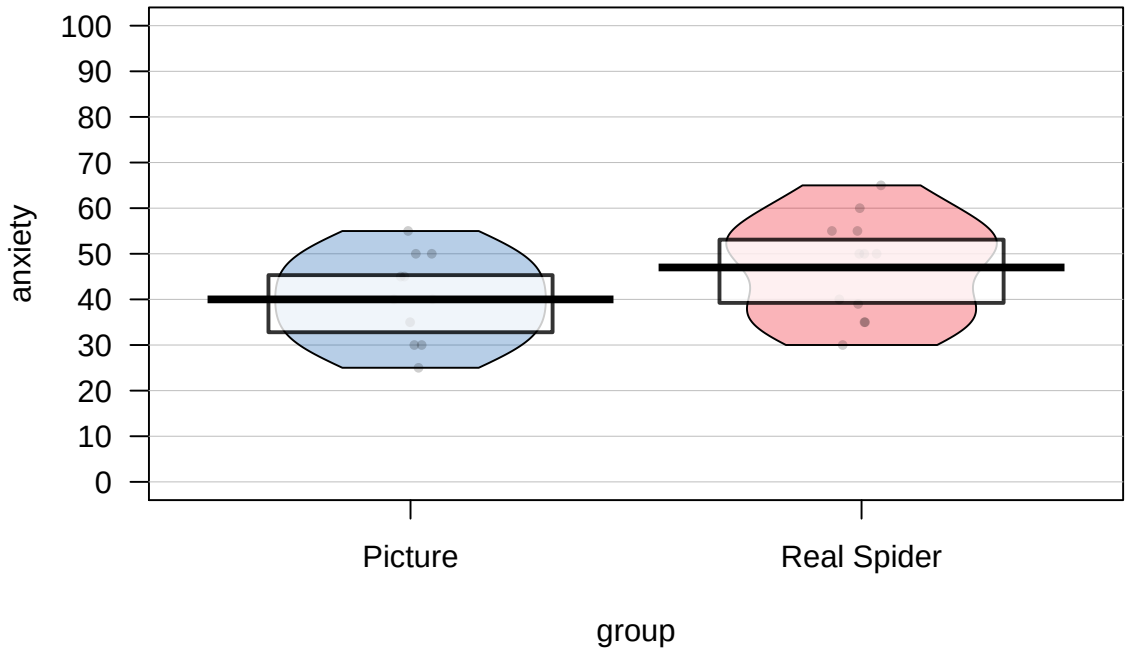


Oder mit einem etwas ausgefalleneren „Pirateplot“:

```
install.packages("yarrr", dep = TRUE)
```

```
library(yarrr)
```

```
pirateplot(anxiety ~ group,
           data = spider_long,
           ylim = c(0, 100))
```



Annahmen des Zweistichproben t-Tests: Zwei unabhängige Stichproben, Normalverteilung

2.1.3 Abhängiger t-Test

Der `spider_long` Datensatz enthält Daten eines between-groups Experiments. `spider_wide` enthält hingegen Ergebnisse eines within-person Experiments: Jeder Proband wurde einmal einer echten Spinne und einmal einem Bild einer Spinne ausgesetzt. Im Anschluss wurde jeweils das Angstniveau gemessen.

Für derartige Daten benötigen wir den abhängigen t-Test. Diesen bekommen wir, indem wir in der Funktion `t.test()` die Option `paired` auf `TRUE` setzen:

```
t.test(spider_wide$real, spider_wide$picture, paired = TRUE)
```

```
##
##  Paired t-test
##
## data:  spider_wide$real and spider_wide$picture
## t = 2.4725, df = 11, p-value = 0.03098
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   0.7687815 13.2312185
```

```
## sample estimates:
## mean of the differences
##                       7
```

Wir sehen, dass die Differenz der Mittelwerte wie beim Zweistichproben-t-Test 7 Punkte beträgt. Dieses Mal ist der Test jedoch signifikant. Das heißt, wir können die Nullhypothese (es gibt keine Unterschiede) verwerfen. Das Ergebnis verdeutlicht also auch sehr eindrücklich, die Überlegenheit von within-person Experimenten gegenüber between-groups Experimenten (zumindest in Hinblick auf die statistische Power).

Annahmen für den abhängigen t-Test: Zwei abhängige Stichproben, normalverteilte Differenzen

2.1.4 Annahmen testen

Der t-Test nimmt an, dass die Werte der unabhängigen Stichproben (beim Zweistichproben-t-Test), beziehungsweise die Differenzen der Messungen (beim abhängigen t-Test) normalverteilt sind. Die Variable muss außerdem Intervallskalenniveau besitzen.

Es wird außerdem oftmals angegeben, dass der t-Test voraussetzt, dass die beiden Gruppen gleiche Varianzen besitzen. Hierzu unten mehr.

2.1.4.1 Test auf Normalverteilung

Der einfachste Weg, um zu testen, ob eine Variable normalverteilt ist, ist der Shapiro-Wilk Test. `R` hält hierfür die Funktion `shapiro.test()` bereit.

```
shapiro.test(spider_long$anxiety)
```

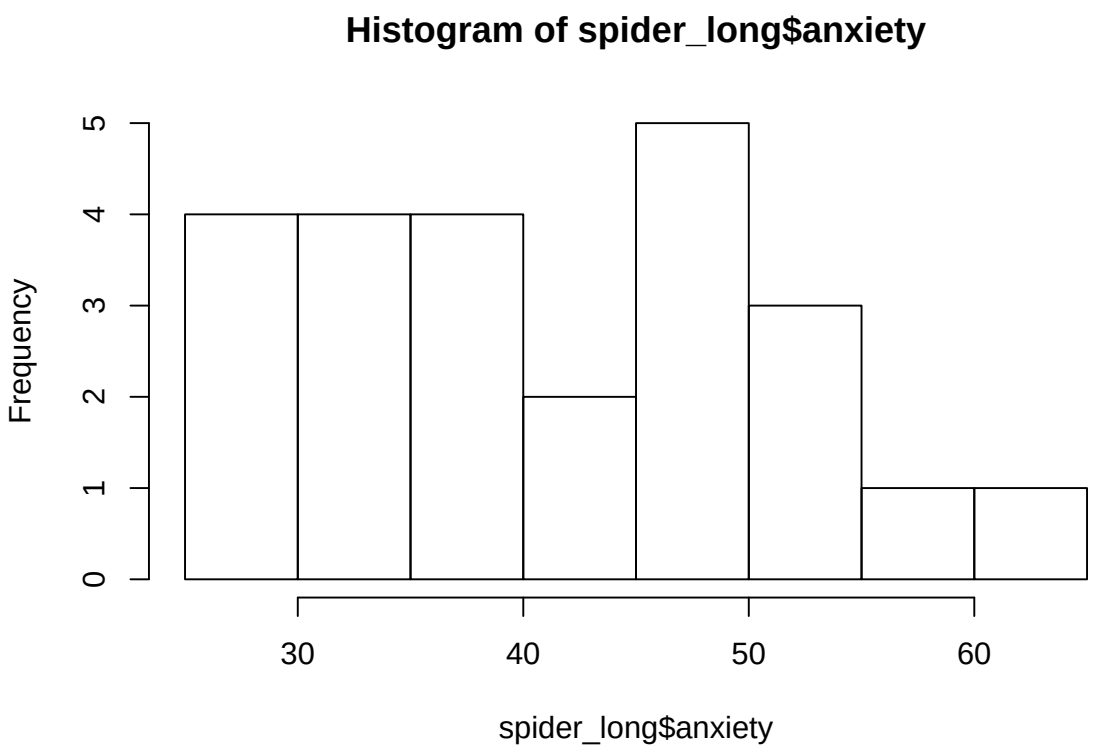
```
##
##  Shapiro-Wilk normality test
##
## data:  spider_long$anxiety
## W = 0.96282, p-value = 0.4977
```

Der Test prüft, ob eine Verteilung signifikant von einer Normalverteilung abweicht. Ein signifikantes Ergebnis bedeutet also, dass eine Variable nicht normalverteilt ist.

Der Shapiro-Wilk Test hat allerdings auch einige Probleme (zum Beispiel, dass er bei großen Samples sehr sensibel ist). Er sollte daher nie als alleiniges Entscheidungskriterium genutzt werden.

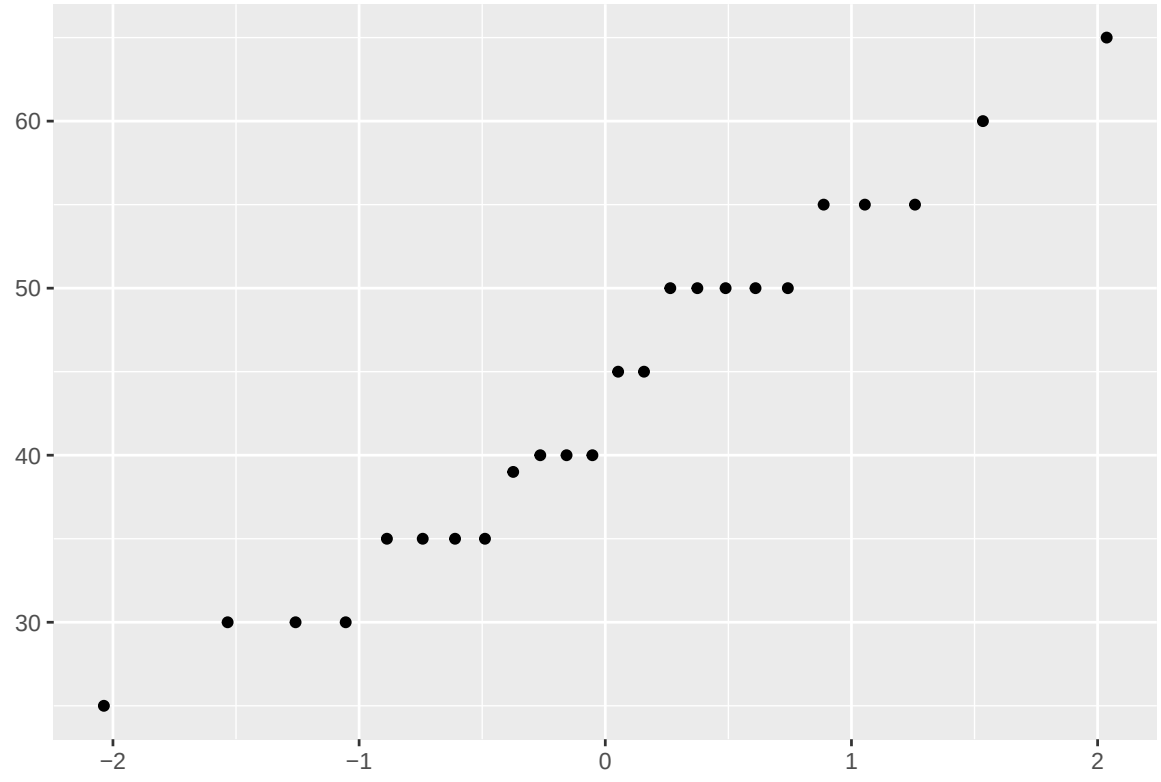
Wir nutzen zusätzlich zum Shapiro-Wilk Test daher graphische Tests. Der einfachste ist der Blick auf das Histogramm.

```
hist(spider_long$anxiety)
```



Das sieht noch nicht so richtig normalverteilt aus. Aber unsere Fallzahl ist auch sehr gering. Nutzen wir noch einen weiteren Test: den Q-Q-Plot, der die Quantile einer Variable gegen die Quantile der Normalverteilung plottet. Hierfür benötigen wir die Funktion `qplot()` aus dem Paket `ggplot2`. Dieses Paket ist ein Teil von `tidyverse`. Wir haben es also bereits geladen.

```
qplot(sample = spider_long$anxiety)
```



Die Punkte sollten bei einer Normalverteilung direkt auf der Diagonalen verlaufen. Verlaufen die Punkte hauptsächlich unter- oder oberhalb der Diagonalen oder in einer S-Form, so liegt vermutlich keine Normalverteilung vor. Hier sieht die Verteilung sehr diagonal aus. Wir vertrauen also der Kombination aus Shapiro-Wilk Test und Q-Q-Plot und nehmen eine Normalverteilung an.

2.1.4.2 Gleiche Varianzen

Die Frage der gleichen Varianzen beider Gruppen als Voraussetzung für einen t-Test ist ein strittiger Punkt. Ich zitiere hier einmal Field et al. (2012: 373):

> „You might have read about homogeneity of variance as being an assumption that is made by the independent t-test. It is the same assumption that we came across in regression, as the homoscedasticity assumption, and statisticians used to recommend testing for it (using Levene's test) and if the assumption was violated, use an adjustment to correct for it. However, more recently statisticians have stopped using this approach, for two reasons. First, violating this assumption only matters if you have unequal group sizes; if you don't have unequal group sizes, the assumption is pretty much irrelevant and can be ignored. Second, the tests of homogeneity of variance tend to work very well when you have equal group sizes and large samples (when it doesn't matter as much if you have violated the assumption) and don't work as well with unequal group sizes and smaller samples – which is exactly when it matters. Plus, there is an adjustment (called Welch's t-test) which is able to correct for violation of this assumption – it's quite hard to do if you have to do it by hand, but very easy to do if you have a computer. If you have violated the assumption, a correction is made – and if you haven't violated the assumption, a correction is not made, so you might as well always do Welch's t-test and forget about the assumption. If you're really interested in this, I like the article by Zimmerman (2004).“

Wie bereits von Field et al. angedeutet, nimmt `R` automatisch eine Korrektur vor, wenn die Varianzen nicht gleich sind. Wir müssen uns daher um diesen Punkt keine Gedanken machen. Wir haben den angepassten t-Test (Welchs t-Test) sogar schon genutzt. Schauen wir noch einmal auf den Output unseres Zweistichproben t-Tests:

```r
t.test(anxiety ~ group, data = spider_long, paired = FALSE)
```

```
##
##  Welch Two Sample t-test
##
## data:  anxiety by group
## t = -1.6813, df = 21.385, p-value = 0.1072
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -15.648641   1.648641
## sample estimates:
##     mean in group Picture mean in group Real Spider
##                        40                        47
```

In der ersten Zeile gibt `R` an, dass der Welch Test genutzt wurde: `Welch Two Sample t-test`. Die Anpassung sieht man auch an der Zahl der Freiheitsgrade (df), die eigentlich 22 betragen müsste ($df = N_1 + N_2 - 2 = 12 + 12 - 2 = 22$). Durch die Anpassung wird der Test „konservativer", d.h. der p-Wert steigt etwas an.

2.2 ANOVA

Um mehr als zwei Gruppen miteinander zu vergleichen, benötigen wir einen anderen Test als den t-Test. Hier kommt *Analysis of Variance* (ANOVA) ins Spiel. ANOVA testet, ob die Mittelwerte von drei oder mehr Gruppen identisch sind.

$$\textit{Nullhypothese } H_0 : \text{Mittelwert der Gruppen sind gleich } (\bar{X}_1 = \bar{X}_2 = \bar{X}_3)$$

$$\textit{Alternativhypothese } H_a : \text{Mittelwerte der Gruppen sind nicht gleich}$$
$$(\bar{X}_1 \neq \bar{X}_2 \neq \bar{X}_3 \text{ ODER } \bar{X}_1 \neq \bar{X}_2 = \bar{X}_3 \text{ ODER } \bar{X}_1 = \bar{X}_2 \neq \bar{X}_3 \text{ ODER } \bar{X}_1 = \bar{X}_3 \neq \bar{X}_2)$$

Der F-Test (das Resultat der ANOVA) ist ein s.g. *omnibus Test*, d.h. er testet einen Gesamteffekt. Wenn das Ergebnis signifikant ist, bedeutet dies, dass die Mittelwerte der Gruppen verschieden sind. Der Test sagt aber nichts darüber, welche Mittelwerte betroffen sind.

Wir laden mal wieder einen neuen Datensatz und bedienen uns dabei erneut bei Field et al. (2012). Der Datensatz enthält Daten eines medizinischen Experiments. Getestet wurde der Einfluss eines Medikaments auf die Gesundheit der Versuchspersonen. Die Probanden haben entweder ein Placebo, eine geringe Dosis des Medikaments oder eine hohe Dosis des Medikaments erhalten. Der Gesundheitszustand spiegelt einen objektiven Messwert wieder (Skala 1–10).

```r
drug <- read_csv("data/drug.csv")
```

```
## Parsed with column specification:
## cols(
##   person = col_double(),
##   dose = col_double(),
##   health = col_double()
## )
```

```
drug
```

```
## # A tibble: 15 x 3
##    person  dose health
##     <dbl> <dbl>  <dbl>
##  1      1     1      3
##  2      2     1      2
##  3      3     1      1
##  4      4     1      1
##  5      5     1      4
##  6      6     2      5
##  7      7     2      2
##  8      8     2      4
##  9      9     2      2
## 10     10     2      3
## 11     11     3      7
## 12     12     3      4
## 13     13     3      5
## 14     14     3      3
## 15     15     3      6
```

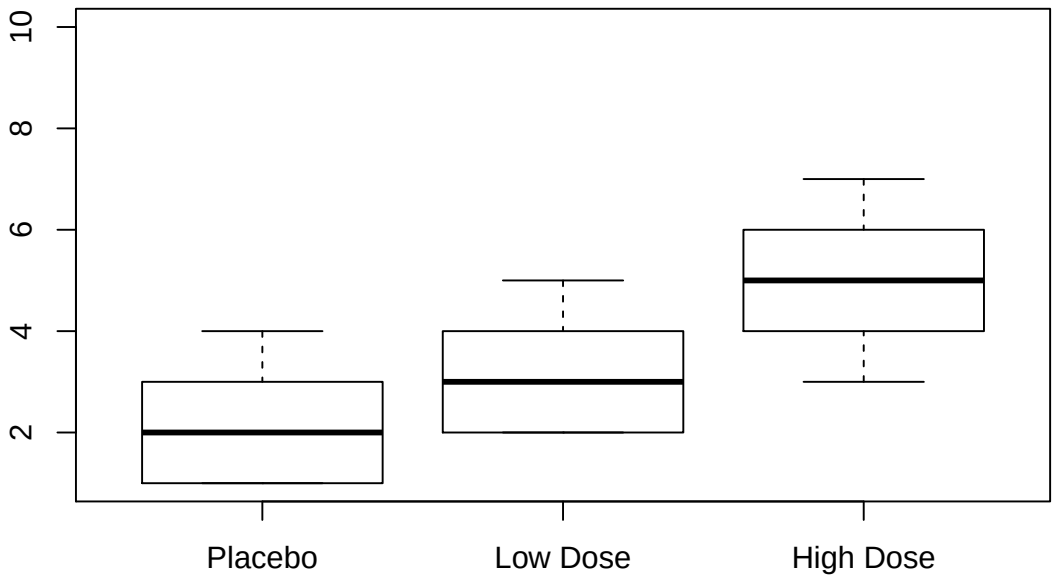
Die Variable `dose` hat die drei Ausprägungen Placebo (1), Low Dose (2) und High Dose (3). Um die Auswertung etwas übersichtlicher zu machen, generieren wir einen Faktor hierfür. Ein Faktor ist in `R` eine besondere Variante einer Variablen, die verdeutlicht, dass es sich um eine kategoriale Variable handelt.

```
drug <- mutate(drug, dose_f = factor(drug$dose,
                                     labels = c("Placebo",
                                                "Low Dose",
                                                "High Dose")))
drug
```

```
## # A tibble: 15 x 4
##    person  dose health dose_f
##     <dbl> <dbl>  <dbl> <fct>
##  1      1     1      3 Placebo
##  2      2     1      2 Placebo
##  3      3     1      1 Placebo
##  4      4     1      1 Placebo
##  5      5     1      4 Placebo
##  6      6     2      5 Low Dose
##  7      7     2      2 Low Dose
##  8      8     2      4 Low Dose
##  9      9     2      2 Low Dose
## 10     10     2      3 Low Dose
## 11     11     3      7 High Dose
## 12     12     3      4 High Dose
## 13     13     3      5 High Dose
## 14     14     3      3 High Dose
## 15     15     3      6 High Dose
```

Ein kurzer Blick auf die Daten

```
boxplot(health ~ dose_f,
        data = drug,
        ylim = c(1, 10))
```



Für ein paar Kennwerte nutzen wir erneut die Funktion `tapply()`, die eine Funktion getrennt nach Gruppen ausführt. Der grundlegende Befehl lautet `tapply(Variable, Gruppe, Output)`

```
tapply(drug$health, drug$dose_f, summary)
```

```
## $Placebo
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0     1.0     2.0     2.2     3.0     4.0
##
## $`Low Dose`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     2.0     2.0     3.0     3.2     4.0     5.0
##
## $`High Dose`
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       3       4       5       5       6       7
```

Vor der ANOVA müssen wir noch testen, ob die Varianzen der Gruppen gleich sind. Bei der ANOVA ist dies wichtiger als beim *t*-Test und `R` passt den Test auch nicht automatisch an. Wir nutzen also zunächst den Levene's Test. Dieser kann mit der Funktion `leveneTest()` aus dem Paket `car` angewendet werden.

```
install.packages("car", dep = TRUE)
```

```r
library(car)
```

```r
leveneTest(health ~ dose_f, data = drug, center = median)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  2  0.1176   0.89
##       12
```

Der Test ist nicht signifikant, das heißt wir müssen die Nullhypothese (gleiche Varianzen) nicht verwerfen. Die ANOVA kann problemlos durchgeführt werden.

```r
drug_anova <- aov(health ~ dose_f, data = drug)
summary(drug_anova)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## dose_f       2  20.13  10.067   5.119 0.0247 *
## Residuals   12  23.60   1.967
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

VORSICHT: Wenn die Gruppierungsvariable (`drug_f`) kein Faktor ist, erkennt `aov()` nicht, dass es sich um mehrere Gruppen handelt und liefert falsche Ergebnisse. Wir können dies mit `drug` testen:

```r
drug_anova2 <- aov(health ~ dose, data = drug)
summary(drug_anova2)
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## dose         1  19.60  19.600   10.56 0.00634 **
## Residuals   13  24.13   1.856
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Am einfachsten erkennt man den Fehler daran, dass die Freiheitsgrade (`Df`) 1 betragen, obwohl wir bei einer ANOVA ein Freiheitsgrad weniger als die Anzahl der Gruppen haben sollten.

Das Ergebnis der ANOVA wird übrigens so berichtet: $F(2{,}12) = 5.12$, $p = .025$

Wenn die Varianzen nicht gleich sind (der Levene's Test ist signifikant), kann man auch hier Welchs F-Test statt des normalen Tests nutzen:

```r
oneway.test(health ~ dose_f, data = drug)
```

```
##
##  One-way analysis of means (not assuming equal variances)
##
## data:  health and dose_f
## F = 4.3205, num df = 2.0000, denom df = 7.9434, p-value = 0.05374
```

Wir sehen, dass die Freiheitsgrade (df) von 12 auf 7,9 angepasst wurden. Der p-Wert fällt daher auch leicht aus dem Bereich von $p < .05$.

2.2.1 Post-hoc Tests

Um nun herauszufinden, welche Gruppen sich signifikant voneinander unterscheiden, könnten wir für jede Kombination einen t-Test durchführen: Placebo vs. Low Dose, Placebo vs. High Dose, Low Dose vs. High Dose. Damit würden wir allerdings die Wahrscheinlichkeit eines Fehlers 1. Art (wir nehmen einen Unterschied an, obwohl es keinen gibt) stark erhöhen. 95 % Wahrscheinlichkeit keinen Fehler 1. Art zu begehen bei drei Tests ergibt $0{,}95^3 = 0{,}857 = 14{,}3$ % (mehr dazu in Field et al. 2012: Section 10.2.1).

Wir müssen daher den p-Wert anpassen. Hierfür gibt es verschiedene Verfahren (s. Field et al. 2012: Section 10.5). Das bekannteste ist die *Bonferroni correction*. Sie teilt den p-Wert schlicht durch die Anzahl der Tests. Bei drei t-Tests also $0,05/3 = 0,017$.

Die Rechenarbeit übernimmt glücklicherweise `R` für uns:

```
pairwise.t.test(drug$health, drug$dose_f,
                paired = FALSE,
                p.adjust.method = "bonferroni")
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  drug$health and drug$dose_f
##
##           Placebo Low Dose
## Low Dose  0.845   -
## High Dose 0.025   0.196
##
## P value adjustment method: bonferroni
```

Wir sehen nun, dass sich lediglich High Dose und Placebo signifikant voneinander unterscheiden ($p < .05$).

2.2.2 Planned Contrasts (Nerd-Content)

Wenn wir bereits vor der ANOVA Annahmen über die Unterschiede haben, sollte man eher *planned contrasts* verwenden. Das Thema ist relativ komplex und ich empfehle daher die Lektüre von Field et al. 2012: 414ff.

Im Grunde geht man so vor, dass man immer eine Gruppe gegen den Rest vergleicht und die Vergleichsgruppe im weiteren Verlauf nicht mehr genutzt wird. Bei drei Gruppen vergleicht man also Gruppe 1 mit Gruppe 2 & 3 und anschließend Gruppe 2 mit Gruppe 3. In unserem Medikamentenbeispiel wäre es am sinnvollsten, die Kontrollgruppe gegen die beiden Treatmentgruppen und dann die Treatmentgruppen untereinander zu vergleichen.

Um einen solchen Vergleich umzusetzen, nutzt man Gewichte (*weights*) für die einzelnen Gruppen. Dabei erhält eine Gruppe negative Werte und die anderen positiven Werte. Nicht genutzte Gruppen erhalten eine 0. Die Summe muss 0 ergeben. In unserem Bespiel sieht das so aus:

Gruppe	Dummy 1	Dummy 2	Product
Placebo	-2	0	0
Low dose	1	-1	-1
High dose	1	1	1
Summe	0	0	0

Die beiden Dummies bauen wir nun in den Datensatz ein.

```
drug <- mutate(drug, contrast1 = ifelse(dose_f == "Placebo", -2, 1))
drug <- mutate(drug, contrast2 = 0)
```

```r
drug <- mutate(drug, contrast2 = ifelse(dose_f == "Low Dose", -1, contrast2))
drug <- mutate(drug, contrast2 = ifelse(dose_f == "High Dose", 1, contrast2))
```

Am Ende rechnen wir nun mit den beiden Dummies eine Regression (hierzu später mehr).

```r
drug_contrast <- lm(health ~ contrast1 + contrast2, data = drug)
summary(drug_contrast)
```

```
##
## Call:
## lm(formula = health ~ contrast1 + contrast2, data = drug)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##   -2.0   -1.2   -0.2    0.9    2.0
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.4667     0.3621   9.574 5.72e-07 ***
## contrast1     0.6333     0.2560   2.474   0.0293 *
## contrast2     0.9000     0.4435   2.029   0.0652 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.402 on 12 degrees of freedom
## Multiple R-squared:  0.4604, Adjusted R-squared:  0.3704
## F-statistic: 5.119 on 2 and 12 DF,  p-value: 0.02469
```

Wir sehen hier, dass nur der Contrast 1 (Treatmentgruppen vs. Kontrollgruppe) signifikant ist. Der Output spiegelt außerdem auch einige Informationen wieder, die wir bereits kennen (könnten). Der Mittelwert aller Beobachtungen beträgt 3,47 (*Intercept*). Die beiden Treatmentgruppen haben einen Mittelwert, der $3 \times 0,63 = 1,9$ über dem der Kontrollgruppe liegt und die High Dose Gruppe hat einen um $2 \times 0,9 = 1,8$ höheren Mittelwert als die Low Dose Gruppe. Das können wir auch leicht überprüfen:

```r
mean(drug$health)
```

```
## [1] 3.466667
```

```r
tapply(drug$health, drug$contrast1, mean)
```

```
##  -2   1
## 2.2 4.1
```

```r
tapply(drug$health, drug$contrast2, mean)
```

```
##  -1   0   1
## 3.2 2.2 5.0
```

```r
# Kontrollgruppe vs. Treatmentgruppen
4.1 - 2.2
```

```
## [1] 1.9
```

```r
(4.1 - 2.2)/3
```

```
## [1] 0.6333333
```

```r
# Low Dose vs- High Dose
5 - 3.2
```

```
## [1] 1.8
```

```r
(5 - 3.2) / 2
```

```
## [1] 0.9
```

2.2.3 Effektstärke

Zum Schluss können wir noch Effektstärken berechnen. Bei einer ANOVA verwendet man üblicherweise Cohen's *f*. Hierbei hilft uns das Paket `sjstats`:

```r
install.packages("sjstats", dep = TRUE)
```

```r
library(sjstats)
```

```r
cohens_f(drug_anova)
```

```
##     term  cohens.f
## 1 dose_f 0.9236381
```

Üblicherwiese werden Werte von 0,1 bis 0,25 als *klein*, zwischen 0,25 und 0,4 als *mittel* und über 0,4 als *groß* betrachtet.

2.3 Korrelation

Um den Zusammenhang zwischen zwei (intervallskalierten) Variablen zu testen, verwendet man in der Regel eine Korrelation. Deren Resultat wird in Form von Pearsons Korrelationskoeffizient r angegeben.

Um eine Korrelation beispielhaft in `R` zu ermitteln, verwenden wir noch einmal den „Lecturer Datensatz" aus Teil 1.

```r
lecturer_data <- read_csv("data/Lecturer_Data.csv")
```

```
## Parsed with column specification:
## cols(
##   ID = col_double(),
##   name = col_character(),
##   birth_date = col_character(),
```

```
##   job = col_double(),
##   friends = col_double(),
##   alcohol = col_double(),
##   income = col_double(),
##   neurotic = col_double()
## )
```

Wir wollen nun wissen, ob es einen Zusammenhang zwischen dem Alkoholkonsum und Neurotizismus gibt. `R` stellt hierfür drei verschiedene Funktionen zur Verfügung: `cor()`, `cor.test()` und `rcorr()` (aus dem Paket `Hmisc`). Die Vor- und Nachteile stellen Field et al. (2012: 216) in folgender Tabelle dar:

Function	Pearson	Spearman	Kendall	p-values	CI	Multiple corr
cor()	Yes	Yes	Yes	No	No	Yes
cor.test()	Yes	Yes	Yes	Yes	Yes	No
rcorr()	Yes	Yes	No	Yes	No	Yes

Die Funktionen unterscheiden sich also zunächst in den Korrelationskoeffizienten, die sie zur Verfügung stellen (Pearsons *r*, Spearmans *rho*, Kendalls *tau*). p-Werte geben nur `cor.test()` und `rcorr()` aus. Konfidenzintervalle gibt es nur mit `cor.test()`. Wenn wir mehrere Variablen paarweise miteinander korrelieren wollen, müssen wir `cor()` oder `rcorr()` verwenden.

Wir entscheiden uns in diesem Fall für `cor.test()`

```
cor.test(lecturer_data$alcohol, lecturer_data$neurotic,
         method = "pearson")
```

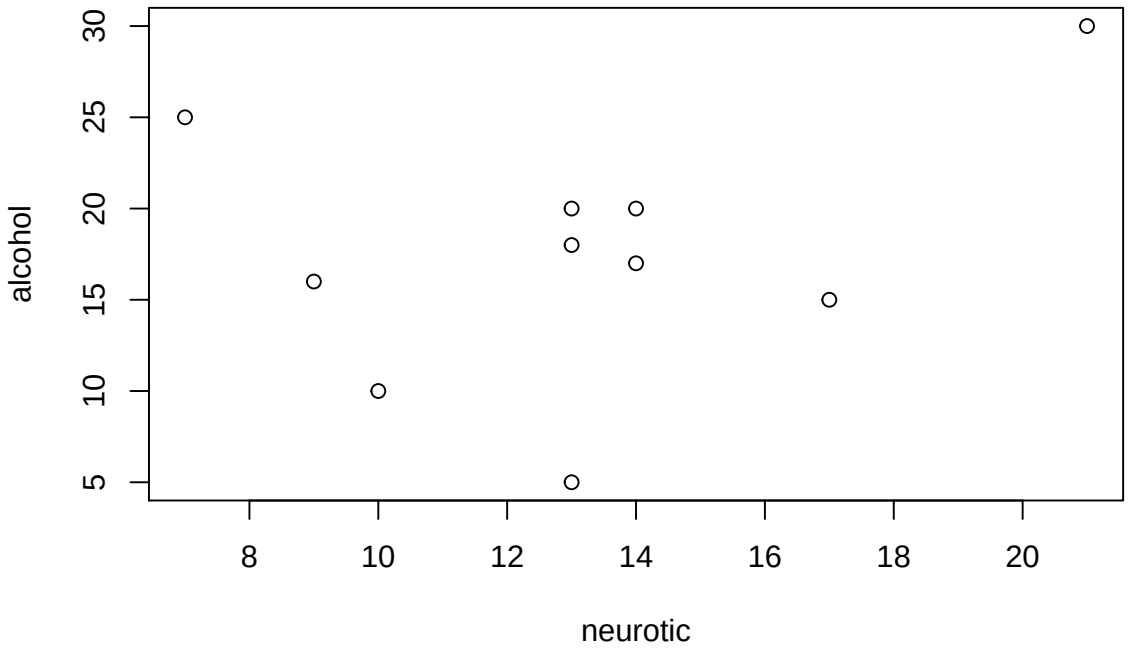
```
##
##  Pearson's product-moment correlation
##
## data:  lecturer_data$alcohol and lecturer_data$neurotic
## t = 0.88472, df = 8, p-value = 0.4021
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.4077319  0.7813026
## sample estimates:
##       cor
## 0.2985335
```

Der Korrelationskoeffizient r beträgt also 0,3. Man könnte von einer mittleren Korrelation sprechen. Allerdings ist der Zusammenhang mit p = 0,40 nicht signifikant.

Auch hier erfolgt das Reporting ähnlich wie bei *t*-Test und ANOVA. Die vollständige Angabe lautet: $r(8) = .30$, $p = .402$

Eine grafische Darstellung erreichen wir zum Beispiel mit der `plot` Funktion.

```
plot(alcohol ~ neurotic, data = lecturer_data)
```



3 Regression

Wir nutzen in diesem Kapitel erneut Daten von Field et al. (2012). Dieses Mal handelt es sich um Daten über Verkaufszahlen von Musikalben (stellen Sie sich einfach vor, wir befinden uns in einer Zeit ohne Streaming).

`album_sales.csv` enthält vier Variablen:

- adverts: Werbeausgaben in Pfund
- sales: Verkaufte Alben in Stück
- airplay: Anzahl der Wiedergabe auf Radio 1 in der Woche vor dem Release
- attract: Attraktivität der Band (basierend auf Umfragen) auf einer Skala von 0 (*hideous potato-heads*) bis 10 (*gorgeous sex objects*)
- hiphop: Handelt es sich um eine Hip-Hop-Band? (0 = nein, 1 = ja)

```
albums <- read_csv("data/album_sales.csv")
```

```
## Parsed with column specification:
## cols(
##   adverts = col_double(),
##   sales = col_double(),
##   airplay = col_double(),
##   attract = col_double(),
##   hiphop = col_double()
## )
```

```
albums
```

```
## # A tibble: 200 x 5
##    adverts  sales airplay attract hiphop
##      <dbl>  <dbl>   <dbl>   <dbl>  <dbl>
##  1     256 330000      43      10      1
##  2  985685 120000      28       7      0
##  3 1445563 360000      35       7      1
##  4 1188193 270000      33       7      1
##  5  574513 220000      44       5      0
##  6  568954 170000      19       5      0
##  7  471814  70000      20       1      1
##  8    8352 210000      22       9      1
##  9  514068 200000      21       7      1
## 10  174093 300000      40       7      1
## # ... with 190 more rows
```

3.1 Bivariate Regression

Versuchen wir zunächst den Erfolg eines Albums (gemessen in verkauften Einheiten) durch den Werbeetat zu erklären und nutzen hierfür ein lineares Regressionsmodell.

`R` stellt hierfür die Funktion `lm()` (linear model) bereit. Ein Befehl hat immer die Form `model_name <- lm(abhängige_variable ~ unabhängige_variable, data = Daten)`

```
model_albums1 <- lm(sales ~ adverts, data = albums)
summary(model_albums1)
```

```
##
## Call:
## lm(formula = sales ~ adverts, data = albums)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -163854  -50351   -3915   46774  430740
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.591e+05  8.512e+03  18.689  < 2e-16 ***
## adverts     6.848e-02  1.230e-02   5.567 8.37e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 79400 on 198 degrees of freedom
## Multiple R-squared:  0.1353, Adjusted R-squared:  0.131
## F-statistic: 30.99 on 1 and 198 DF,  p-value: 8.372e-08
```

Hier finden wir nun alle Informationen, die wir für die Interpretation eines Regressionsmodells benötigen.

Informationen zur Modellgüte:

- R^2 (`Multiple R-squared`): 0.14
- Korrigierter R^2 (`Adjusted R-squared`): 0.13
- F-Statistik einer Varianzanalyse (`F-statistic`):
  - Der erste Wert (30.99) ist der *F*-Wert, das Verhältnis von *mean squares* des Modells (Differenz der mittleren quadrierten Abweichungen vom Mittelwert und der mittleren quadrierten

Abweichung von der Regressionsgeraden) und *mean squares* der Residuen (mittlere quadrierte Abweichung von der Regressionsgeraden) : $F = \frac{MS_M}{MS_R}$
  - Anzahl der genutzten und ungenutzten Freiheitsgrade (`DF`) für den F-Test: 1, 198
  - p-Wert des F-Tests (`p-value`): 8.372e-08 ==> < 0,001 ==> Unser Modell sagt die Zahl der Albumverkäufe signifikant besser vorher als der reine Mittelwert der Verkäufe.

Informationen zu den Modellparametern

Wir haben zwei Modellparameter:

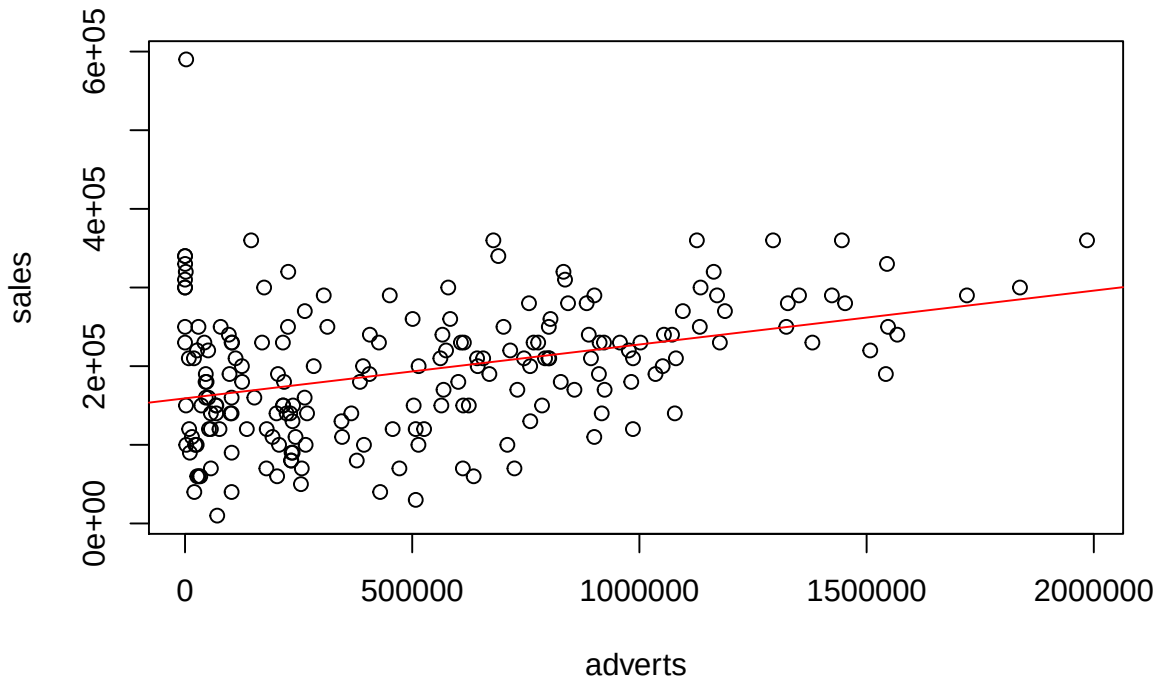
- Die Konstante (`(Intercept)`), d.h. der Schnittpunkt mit der y-Achse
- Die Werbeausgaben (`adverts`)

Für jeden Parameter haben wir verschiedene Informationen:

- Koeffizienten (`Estimate`)
  - Konstante: Ein Album ohne Werbeausgaben verkauft sich laut Modell 159083.7 mal.
  - Werbeausgaben: Mit jedem zusätzlichen Pfund an Werbeausgaben steigt die Zahl der verkauften Exemplare um 0.0685.
- Standardfehler (`Std. Error`) der Koeffizienten: 8512.13 / 0.01230
- t-Werte für die Koeffizienten (`t value`): 18.69 / 5.57
- p-Wert des t-Tests (`Pr(>|t|)`): < 0.001 / < 0.001 ==> Der Koeffizient unterscheidet sich in beiden Fällen signifikant von Null.

Wir können unser schönes Modell auch grafisch betrachten:

```
plot(sales ~ adverts, data = albums)
abline(model_albums1, col = "red")
```



3.2 Multiple Regression

Die bivariate Regression ist in `R` einfach zu einer multiplen Regression ergänzt. Weitere unabhängige Variablen werden einfach mit `+` angehängt.

Fügen wir also zu unserem Modell noch die Anzahl der Wiedergaben auf Radio 1 und die Attraktivität der Band hinzu:

```
model_albums2 <- lm(sales ~ adverts + airplay + attract, data = albums)
summary(model_albums2)
```

```
##
## Call:
## lm(formula = sales ~ adverts + airplay + attract, data = albums)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -156799  -35019   -4632   30155  299239
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.253e+04  2.222e+04  -1.464    0.145
## adverts      5.664e-02  9.365e-03   6.048 7.29e-09 ***
## airplay      3.663e+03  3.551e+02  10.316  < 2e-16 ***
## attract      1.433e+04  3.104e+03   4.618 7.01e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 60070 on 196 degrees of freedom
## Multiple R-squared:   0.51,  Adjusted R-squared:  0.5025
## F-statistic:    68 on 3 and 196 DF,  p-value: < 2.2e-16
```

Wir sehen, dass sich das Modell deutlich verbessert hat. Der korrigierte R^2 ist von 0.13 auf 0.5 angestiegen. Alle drei Variablen haben einen signifikanten Einfluss auf die Verkaufszahlen eines Albums. Doch welche hat den größten Einfluss? Das lässt sich so einfach nicht beantworten, da alle Variablen unterschiedlich skaliert sind. Wir benötigen daher standardisierte Regressionskoeffizienten. Um diese zu bekommen, benötigen wir die Funktion `lm.beta()` aus dem Paket `lm.beta`.

```
install.packages("lm.beta", dep = TRUE)
```

```
library(lm.beta)
```

```r
model_albums2_beta <- lm.beta(model_albums2)
summary(model_albums2_beta)
```

```
##
## Call:
## lm(formula = sales ~ adverts + airplay + attract, data = albums)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -156799  -35019   -4632   30155  299239
##
## Coefficients:
##              Estimate Standardized Std. Error t value Pr(>|t|)
## (Intercept) -3.253e+04    0.000e+00  2.222e+04  -1.464    0.145
## adverts      5.664e-02    3.043e-01  9.365e-03   6.048 7.29e-09 ***
## airplay      3.663e+03    5.277e-01  3.551e+02  10.316  < 2e-16 ***
## attract      1.433e+04    2.348e-01  3.104e+03   4.618 7.01e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 60070 on 196 degrees of freedom
## Multiple R-squared:   0.51,  Adjusted R-squared:  0.5025
## F-statistic:    68 on 3 and 196 DF,  p-value: < 2.2e-16
```

Die Zahl der Wiedergaben in Radio 1 ist also am einflussreichsten. Mit einer Zunahme der Wiedergaben auf Radio 1 um eine Standardabweichung (= 12.27) steigt die Zahl der verkauften Alben um 0.53 Standardabweichungen (= 0.53 x 85171.48 = 44947.00).

3.3 Regression mit kategorialen Daten

3.3.1 Dummies

Die lineare Regression kann nur mit intervallskalierten Variablen genutzt werden. Die einzige Ausnahme hiervon sind dichotome Variablen (Variablen mit zwei Ausprägungen). Diese können als unabhängige Variablen in einer Regression eingesetzt werden. Man nennt diese Variablen auch Dummy-Variablen. Es hat sich etabliert, dass diese mit 0 und 1 codiert werden. Handelt es sich um Ja/Nein-Variablen, so wird Nein in der Regel mit 0 codiert. So wäre ein Dummy, der angibt, ob eine Person Führungskraft ist, mit 0 = nein und 1 = ja codiert.

Die Interpretation des Regressionskoeffizienten eines Dummies ist relativ einfach. Er gibt an, wie sich die abhängige Variable ändert, wenn der Dummy „von 0 nach 1 wechselt“. Ein Beispiel mit den bereits verwendeten Albumdaten. Der Datensatz enthält die Variable `hiphop`, die angibt, ob es sich bei der Band um eine Hip-Hop-Band handelt oder nicht (0 = nein; 1 = ja). Wir wollen wissen, ob sich Hip-Hop-Band besser verkaufen als andere Bands:

```r
# Regressionsmodell spezifizieren
model_albums_hiphop <- lm(sales ~ adverts + airplay + attract + hiphop, data = albums)

# Ergebnisse
summary(model_albums_hiphop)
```

```
##
## Call:
## lm(formula = sales ~ adverts + airplay + attract + hiphop, data = albums)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -149324  -34253     503   35399  292629
```

```
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.289e+04  2.177e+04  -1.511  0.13241
## adverts      5.389e-02  9.218e-03   5.846 2.09e-08 ***
## airplay      3.605e+03  3.484e+02  10.348  < 2e-16 ***
## attract      1.250e+04  3.099e+03   4.033 7.90e-05 ***
## hiphop       2.657e+04  8.726e+03   3.045  0.00265 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 58840 on 195 degrees of freedom
## Multiple R-squared:  0.5323, Adjusted R-squared:  0.5227
## F-statistic: 55.47 on 4 and 195 DF,  p-value: < 2.2e-16
```

Wir sehen, dass der Hip-Hop-Dummy signifikant ist und einen Koeffizienten von 26573.6 aufweist. Dies bedeutet, dass Hip-Hop-Bands (`hiphop` = 1) im Durchschnitt bei gleichen Werbeausgaben, Wiedergaben im Radio und gleicher Attraktivität 26573.6 mehr Alben verkaufen als andere Bands (`hiphop` = 0).

3.3.2 Kategoriale Variablen

Die Möglichkeit, Dummies in der Regression einzusetzen, können wir uns auch für kategoriale Variablen zu Nutze machen. Man kann für jede Kategorie einen Dummy bilden und diese in der Regression verwenden. Einer der Dummies muss allerdings ausgelassen werden. Er fungiert als Referenzkategorie für die Interpretation der übrigen Dummies.

Illustrieren wird dies einmal anhand von *paycat* (die Lohngruppe) aus dem Federal Employee Viewpoint Survey.

```
# Daten importieren
fevs <- read_csv("data/fevs_2014_subsample.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   agency = col_character(),
##   plevel1 = col_character(),
##   plevel2 = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
table(fevs$paycat)
```

```
##
##   1   2   3
## 461 458 182
```

Die Variable hat drei Ausprägungen. Diese werden wir nun in drei Dummies überführen (paycat1, paycat2, paycat3).

```
fevs <- mutate(fevs, paycat1 = ifelse(paycat == 1, 1, 0))
fevs <- mutate(fevs, paycat2 = ifelse(paycat == 2, 1, 0))
fevs <- mutate(fevs, paycat3 = ifelse(paycat == 3, 1, 0))
```

Wir wählen nun die kleinste Lohngruppe als Referenz. Das heißt wir fügen diese nicht in die Regression ein.

```r
# Job Satisfaction generieren
fevs <- mutate(fevs, job_satisfaction = q40 + q69 + q71)

# Regressionsmodell spezifizieren
model_paycat <- lm(job_satisfaction ~ paycat2 + paycat3, data = fevs)
```

```r
# Regressionsergebnis anzeigen
summary(model_paycat)
```

```
##
## Call:
## lm(formula = job_satisfaction ~ paycat2 + paycat3, data = fevs)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.8508 -1.8508  0.6836  2.1492  4.6836
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  10.3164     0.1421  72.616  < 2e-16 ***
## paycat2       0.5344     0.2012   2.655  0.00804 **
## paycat3       0.5028     0.2678   1.878  0.06071 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.02 on 1075 degrees of freedom
##   (100 observations deleted due to missingness)
## Multiple R-squared:  0.007349,   Adjusted R-squared:  0.005502
## F-statistic: 3.979 on 2 and 1075 DF,  p-value: 0.01897
```

Wir sehen nun, dass die Beschäftigten der Lohngruppe 2 durchschnittlich eine um 0.53 Punkte höhere Arbeitszufriedenheit haben als diejenigen der Lohngruppe 1 (Referenzkategorie). Bei Lohngruppe 3 beträgt der Unterschied 0.5 Punkte im Vergleich zur Lohngruppe 1 (auch wenn dieser Effekt nicht signifikant ist).

3.3.3 Moderationseffekte

Mit Hilfe der linearen Regression lassen sich auch Moderationseffekte testen. Ein Moderationseffekt liegt vor, wenn der Effekt einer unabhängigen Variable von einer anderen Variable abhängt. Im Fall unserer Albumverkäufe könnte der Effekt der Wiedergaben im Radio von der Art der Band abhängen. Oder der Effekt, den die Werbeausgaben haben von der Attraktivität der Band.

Um einen solchen Moderationseffekt zu testen, verwendet man Interaktionsterme. Dies bedeutet, dass die beiden Variablen, die den Moderationseffekt bilden, miteinander multipliziert und dieser Term zusätzlich in die Regressionsgleichung eingefügt wird.

Wir schätzen nun eine Regression zunächst nur mit den verfügbaren unabhängigen Variablen

```r
model1 <- lm(sales ~ adverts + attract + airplay + hiphop, data = albums)
```

```r
summary(model1)
```

```
##
## Call:
## lm(formula = sales ~ adverts + attract + airplay + hiphop, data = albums)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -149324  -34253     503   35399  292629
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.289e+04  2.177e+04  -1.511  0.13241
## adverts      5.389e-02  9.218e-03   5.846 2.09e-08 ***
## attract      1.250e+04  3.099e+03   4.033 7.90e-05 ***
## airplay      3.605e+03  3.484e+02  10.348  < 2e-16 ***
## hiphop       2.657e+04  8.726e+03   3.045  0.00265 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 58840 on 195 degrees of freedom
## Multiple R-squared:  0.5323, Adjusted R-squared:  0.5227
## F-statistic: 55.47 on 4 and 195 DF,  p-value: < 2.2e-16
```

Wir wollen nun testen, ob der Effekt der Wiedergaben im Radio (*airplay*) von der Art der Band (*hiphop*) abhängig ist. Hierzu fügen wir den Interaktionsterm `airplay*hiphop` in das Modell ein. Da `R` automatisch die Haupteffekte (*hiphop* und *airplay*) hinzufügt, müssen wir diese nicht extra angeben.

```
model2 <- lm(sales ~ adverts + attract + airplay*hiphop, data = albums)
summary(model2)
```

```
##
## Call:
## lm(formula = sales ~ adverts + attract + airplay * hiphop, data = albums)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -155390  -31183   -1786   32900  272847
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.435e+03  2.452e+04   0.059  0.95339
## adverts         5.091e-02  9.113e-03   5.586 7.76e-08 ***
## attract         1.251e+04  3.044e+03   4.110 5.84e-05 ***
## airplay         2.335e+03  5.607e+02   4.164 4.71e-05 ***
## hiphop         -2.725e+04  2.068e+04  -1.318  0.18920
## airplay:hiphop  2.010e+03  7.028e+02   2.859  0.00471 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 57790 on 194 degrees of freedom
## Multiple R-squared:  0.5512, Adjusted R-squared:  0.5396
## F-statistic: 47.65 on 5 and 194 DF,  p-value: < 2.2e-16
```

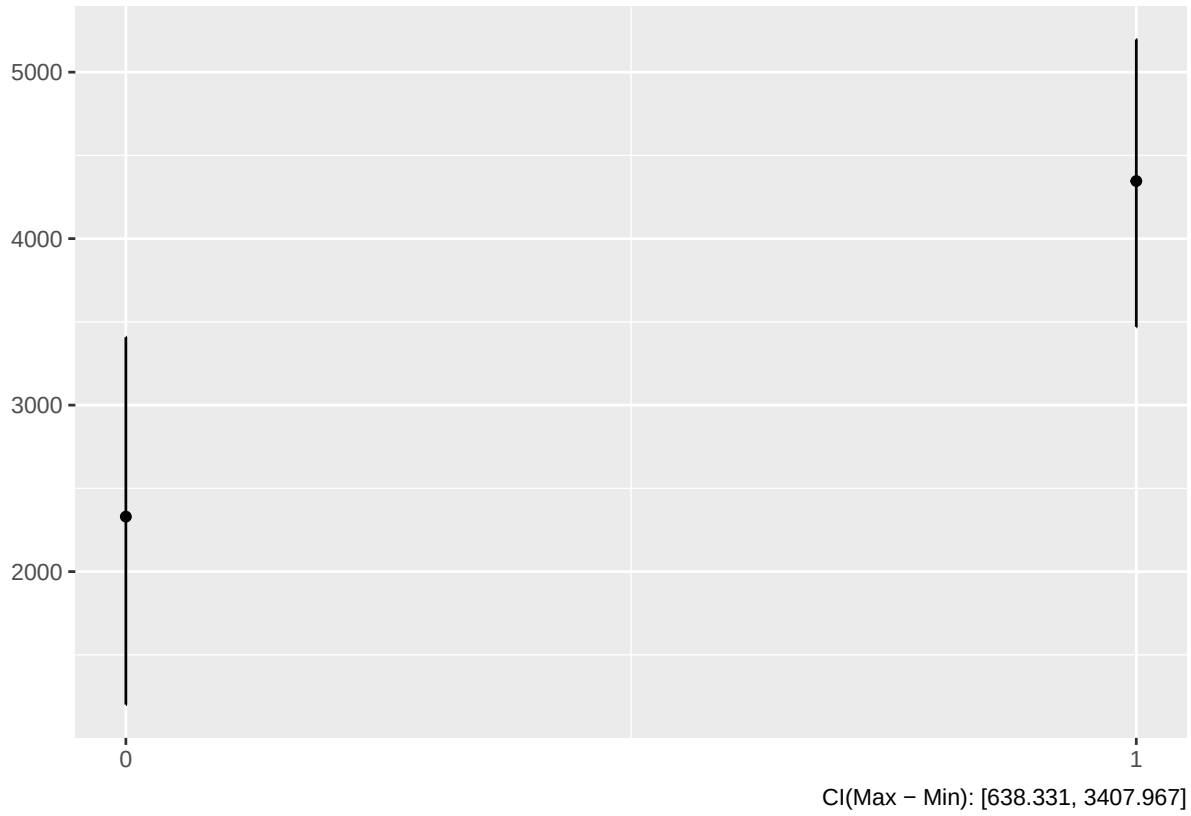
Unter `airplay:hiphop` sehen wir nun den Interaktionsterm. Da dieser signifikant ist, können wir davon ausgehen, dass eine Moderation vorliegt.

Die Moderation können wir auch grafisch darstellen. Hierzu verwenden wir die Funktion `interplot()` aus dem gleichnamigen Paket.

```
install.packages("interplot", dep = TRUE)
```

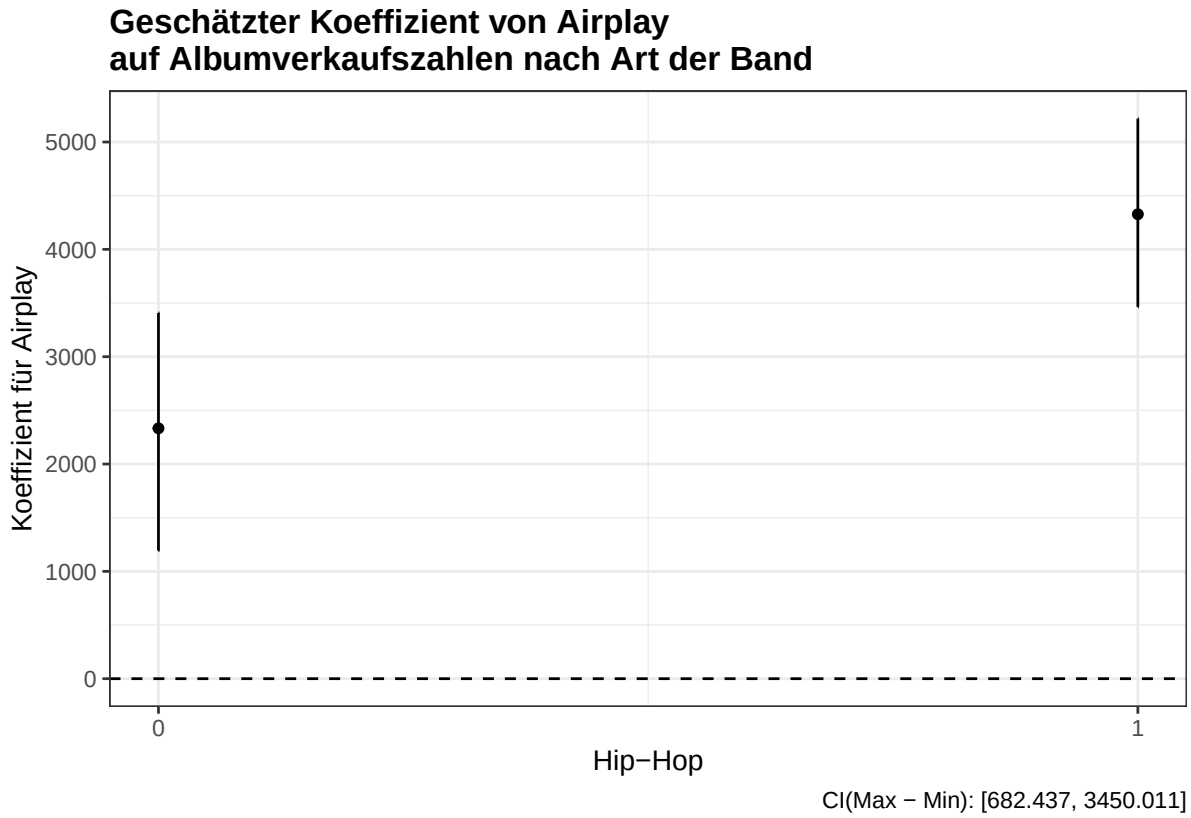
`interplot()` erstellt einen s.g. Marginal Effects Plot. Dieser zeigt auf der y-Achse immer den Effekt der unabhängigen Variable (in unserem Fall *airplay*) auf die abhängige Variable und auf der x-Achse die Werte des Moderators (in unserem Fall *hiphop*).

```
interplot::interplot(m = model2, var1 = "airplay", var2 = "hiphop")
```



Das Ganze können wir auch noch ein bisschen aufhübschen:

```
interplot::interplot(m = model2,
                     var1 = "airplay",
                     var2 = "hiphop",
                     sims = 1000) +
  # Add labels for X and Y axes
  xlab("Hip-Hop") +
  ylab("Koeffizient für Airplay") +
  # Change the background
  theme_bw() +
  # Add the title
  ggtitle("Geschätzter Koeffizient von Airplay \nauf Albumverkaufszahlen nach Art der Band") +
  theme(plot.title = element_text(face="bold")) +
  # Add a horizontal line at y = 0
  geom_hline(yintercept = 0, linetype = "dashed")
```



Wir sehen also, dass der Effekt einer zusätzlichen Widergabe in Radio 1 auf die Verkaufszahlen für Hip-Hop-Bands ca. 4400 und für andere Bands ca. 2400 beträgt.

Ein solcher Moderationseffekt kann nicht nur zwischen einer intervallskalierten und einer Dummyvariable getestet werden, sondern auch zwischen zwei intervallskalierten Variablen. Das Vorgehen hierfür ist dasselbe, nur die Interpretation ist etwas schwieriger.

Nehmen wir als Beispiel einen Moderationseffekt zwischen Werbeausgaben und Attraktivität der Band:

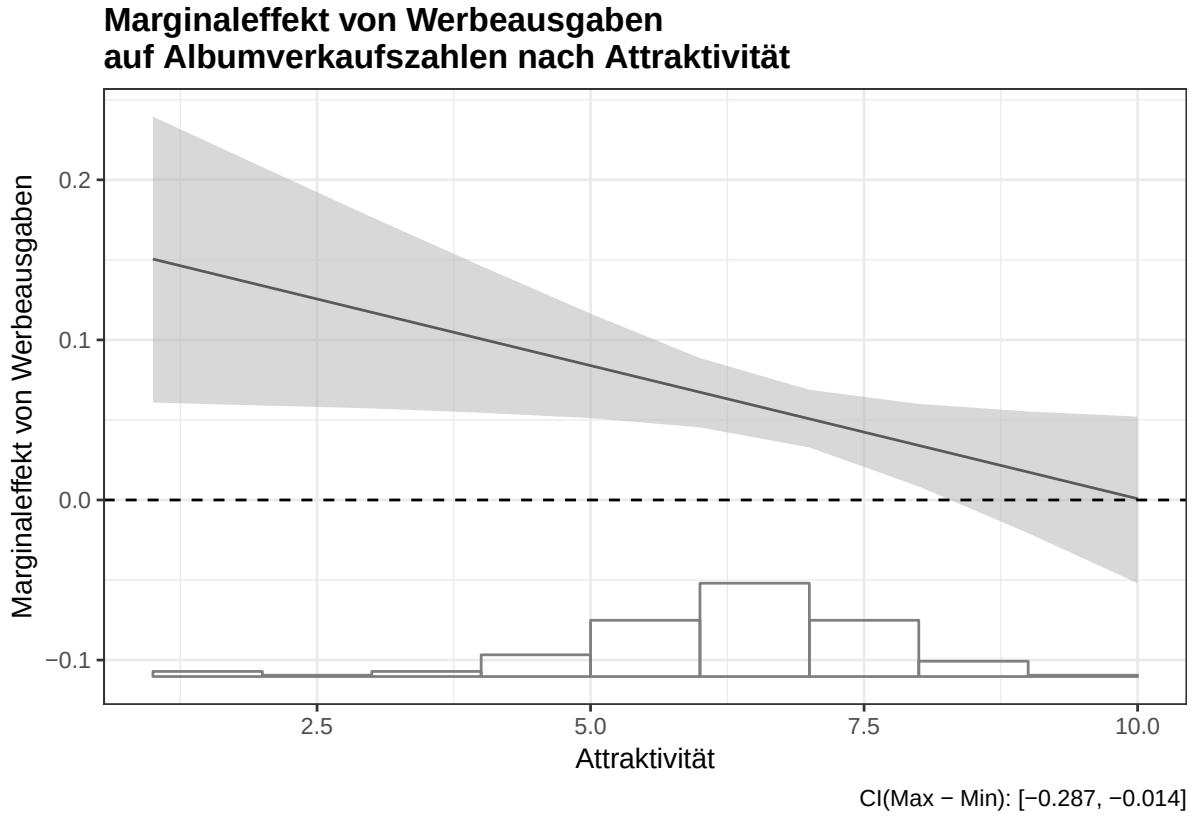
```
model3 <- lm(sales ~ airplay + hiphop + adverts*attract, data = albums)
summary(model3)
```

```
##
## Call:
## lm(formula = sales ~ airplay + hiphop + adverts * attract, data = albums)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -155399  -34070     710   34816  278759
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -7.612e+04  2.895e+04  -2.629  0.00925 **
## airplay          3.583e+03  3.450e+02  10.387  < 2e-16 ***
## hiphop           2.699e+04  8.640e+03   3.124  0.00206 **
## adverts          1.671e-01  5.146e-02   3.247  0.00137 **
## attract          1.893e+04  4.207e+03   4.500 1.17e-05 ***
## adverts:attract -1.666e-02  7.452e-03  -2.235  0.02654 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 58250 on 194 degrees of freedom
```

```
## Multiple R-squared:  0.544,  Adjusted R-squared:  0.5322
## F-statistic: 46.29 on 5 and 194 DF,  p-value: < 2.2e-16
```

Auch hier ist der Interaktionsterm signifikant. Wir schauen uns daher den Marginal Effects Plot an:

```
interplot::interplot(m = model3, var1 = "adverts", var2 = "attract", hist = TRUE) +
  # Add labels for X and Y axes
  xlab("Attraktivität") +
  ylab("Marginaleffekt von Werbeausgaben") +
  # Change the background
  theme_bw() +
  # Add the title
  ggtitle("Marginaleffekt von Werbeausgaben \nauf Albumverkaufszahlen nach Attraktivität") +
  theme(plot.title = element_text(face="bold")) +
  # Add a horizontal line at y = 0
  geom_hline(yintercept = 0, linetype = "dashed")
```



Wir sehen nun den Effekt von Werbeausgaben auf die Verkaufszahlen für die verschiedenen Attraktivitätswerte (1–10). Der Effekt ist folglich umso stärker, je geringer die Attraktivität der Band ist. Für Werte über ca. 8 ist der Effekt jedoch nicht mehr signifikant (der Konfidenzintervall schließt die Null mit ein). Das Histogramm an der x-Achse zeigt uns außerdem, wie häufig die verschiedenen Attraktivitätswerte vorkommen.

3.4 Annahmen der linearen Regression

Die lineare Regression stellt einige Anforderungen an die Daten:

- **Variablentyp**: Intervallskalierte Variablen oder kategoriale Variablen mit zwei Ausprägungen (s.g. Dummies)
- **Varianz**: Die Variablen müssen eine Varianz besitzen, die ungleich Null ist.
- **Keine (perfekte) Multikollinearität** zwischen unabhängigen Variablen
- **Unabhängigkeit von externen Variablen**: Unabhängige Variablen sind nicht mit „externen (unbeobachteten) Variablen“ korreliert.
- **Homoskedastizität**: Die Varianz der Residuen soll für alle Werte der unabhängige(n) Variable(n) gleich sein.
- **Unabhängige Fehler**: Die Residuen aller Beobachtungen sollen unabhängig voneinander sein.
- **Normalverteilung der Residuen**
- **Unabhängigkeit der Beobachtungen**
- **Linearität**: Es wird angenommen, dass ein linearer Zusammenhang zwischen unabhängigen Variablen und abhängiger Variable besteht.

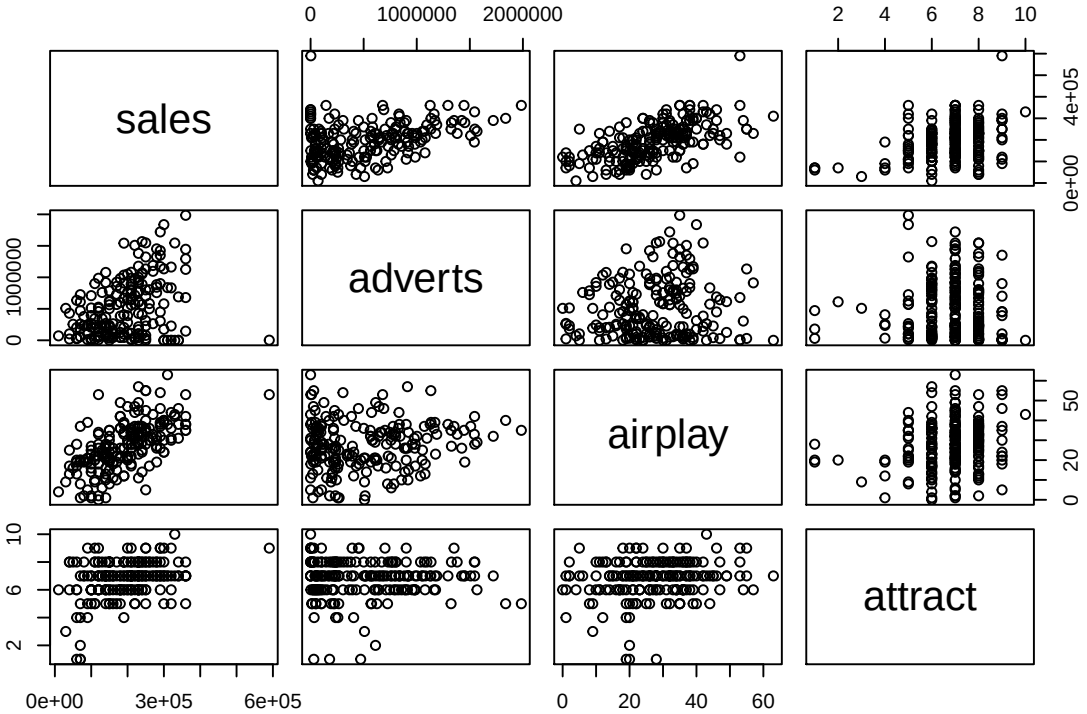
3.5 Regressionsdiagnostik

Fünf der oben genannten Annahmen kann man statistisch oder grafisch testen: Linearität, Multikollinearität, unabhängige Fehler, Homoskedastizität und Normalverteilung der Residuen.

3.5.1 Linearität

Der Zusammenhang zwischen der abhängigen Variable und den unabhängigen Variablen muss linear sein, sonst können wir den Zusammenhang nicht mit einem linearen Regressionsmodell erklären. Die einfachste Variante zum Testen der Linearität ist eine Scatterplot Matrix. Diese stellt den Zusammenhang zwischen jeweils zwei Variablen grafisch dar. In `R` verwenden wir hierfür die Funktion `pairs()`, in die wir einfach die Regressionsgleichung (`sales ~ adverts + airplay + attract`) und den Datensatz eingeben.

```
pairs(sales ~ adverts + airplay + attract, data = albums)
```



In der Matrix interessiert uns vor allem die erste Spalte, die den Zusammenhang zwischen der abhängigen Variable (*Sales*) und den unabhängigen Variablen darstellt. Die drei Grafiken lassen nicht vermuten, dass in einem der Fälle ein nicht-linearer Zusammenhang besteht. Wir können also Linearität annehmen.

3.5.2 Multikollinearität

Das Vorliegen von Multikollinearität kann man mit dem s.g. Variance Inflation Factor (VIF) testen. `vif()` aus dem Paket `car` gibt uns die nötigen Informationen. Wir wenden diese Funktion auf unsere multiple Regression von vorhin an.

```
vif(model_albums2)
```

```
##  adverts  airplay  attract
## 1.012493 1.046708 1.034255
```

Es gibt verschiedene Auffassungen darüber, was problematische VIF-Werte sind. Gemeinhin wird empfohlen, dass Werte über 3 aufmerksam machen sollten. Werte über 9 werden als problematisch betrachtet. Wir haben hier also keine Probleme. Die unabhängigen Variablen sind hinreichend unabhängig voneinander.

3.5.3 Unabhängige Fehler

Die Residuen aller Beobachtungen sollen unabhängig voneinander sein. Wir können den Durbin-Watson Test nutzen, um dies zu testen. Das Paket `car` stellt hierfür die Funktion `durbinWatsonTest()` zur Verfügung.

```
durbinWatsonTest(model_albums2)
```

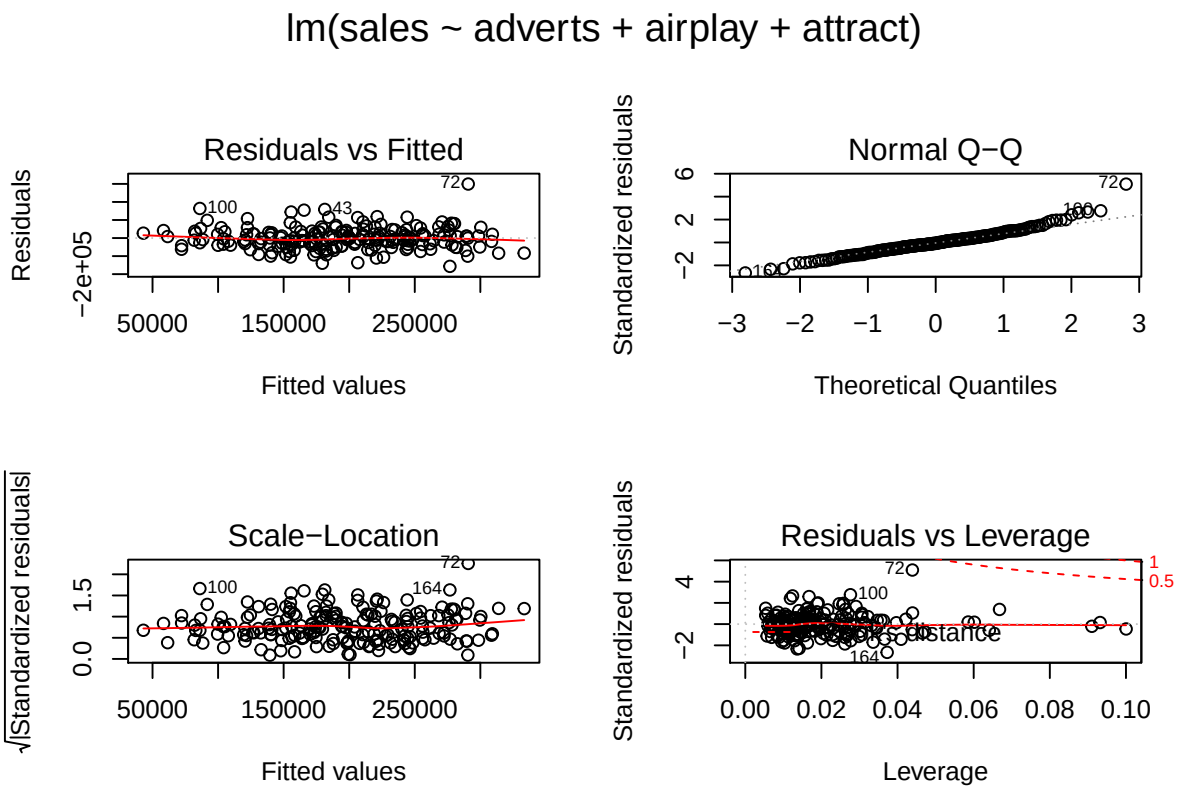
```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.0384975       1.90029   0.488
##  Alternative hypothesis: rho != 0
```

Der `D-W Statistic`-Wert sollte über 1 und unter 3 liegen. Je näher an 2 desto besser. Wir sind hier sehr nahe an 2 und haben folglich kein Problem mit der Unabhängigkeit der Fehler.

3.5.4 Homoskedastizität

Die Frage, ob die Varianz der Residuen über alle Werte der unabhängigen Variablen gleich ist (Homoskedastizität) beantwortet man am besten grafisch. Wir können hierfür das gespeicherte Regressionsmodell (`model_albums2`) an die `plot()` Funktion übergeben:

```
plot(model_albums2)
```



Der erste Graph plottet die Residuen jeder Beobachtung gegen den prognostizierten Wert. Diese Grafik sollte wie eine zufällig Punktwolke aussehen. Ein trichterförmiges Bild deutet auf Probleme hin. Ein kurvenförmiger Verlauf der Punktewolke deutet darauf hin, dass der Zusammenhang zwischen abhängiger und unabhängigen Variablen nicht linear ist.

3.5.5 Normalverteilung der Residuen

Die Grafiken von oben bieten auch eine grafische Überprüfungsmöglichkeit der Normalverteilung. Hierfür findet wieder der bereits erwähnte Q-Q-Plot Anwendung. Die zweite Grafik zeigt einen solchen Plot, der die Quantile der Normalverteilung gegen die Quantile der (standardisierten) Residuen plottet. Die Punkte sollten dabei möglichst linear auf der Diagonalen liegen. Dies ist hier insgesamt gegeben.